EasyRTC with recording command

# 1. EasyRTC Overview



Figure: EasyRTC Overview

The order to take meeting is like following.

a). Through Web Server, each device create communication socket for Peer Connection.

b). Establish Peer Connection with socket created above.

c). Send/Receive Audio/Video eachother.

# 2. Recording Command to record both devices at same time.

## 1). Workflow on sending side.

To send record command other side, use “setPresence” signal. That is, User send this signal other side.

|  |
| --- |
| //easyrtc.js file  sendSignalling(Object.keys(peerConns)[0], "setPresence", { setPresence: { 'show': 'chat', 'status': "record-start" } },  function (msgType, msgData) {  console.log("record-start OK");  },  function (errorCode, errorText) {  console.log(errorText);  }); |

In this here, We must start/stop recording at same time. For this, include “record-start”/”record-end” sub command in “setPresence” signal’s data json.

|  |
| --- |
| //Start recording  { setPresence: { 'show': 'chat', 'status': "record-start" } }  //End recording  { setPresence: { 'show': 'chat', 'status': "record-end" } } |

## 2) Workflow on receiving side.

In receiving side, process “record-start”/”record-end” sub command in “onChannelCm” function.

|  |
| --- |
| var onChannelCmd = function(msg, ackAcceptorFn) {  …  switch (msgType) {  …  case "roomData":  try{  var updateClient = Object.values(msg.msgData.roomData)[0].clientListDelta.updateClient  if( Object.values(updateClient)[0].presence.show == "chat" ){  var status = Object.values(updateClient)[0].presence.status;  switch(status)  {  case "record-start"://Process “record-start” sub-command  if(self.callbackStartRecord) self.callbackStartRecord();  console.log("record-start");  return;  case "record-stop"://Process “record-stop” sub-command  if(self.callbackStopRecord) self.callbackStopRecord();  console.log("record-stop");  return;  }  }  }  catch(e){};  processRoomData(msgData.roomData);  break;  …  }; |